UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/082,794	02/22/2002	David Bau III	109870-130096	2046

25943 7590 02/22/2008

SCHWABE, WILLIAMSON & WYATT, P.C.
PACWEST CENTER, SUITE 1900
1211 SW FIFTH AVENUE
PORTLAND, OR 97204

EXAMINER	
RUTTEN, JAMES D	

ART UNIT	PAPER NUMBER
2192	

MAIL DATE	DELIVERY MODE
02/22/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/082,794	BAU ET AL.	
	Examiner	Art Unit	
	J. Derek Rutten	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *04 December 2007*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3,6-10,12-17,20-22,31-33,35-39 and 42-44* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3,6-10,12-17,20-22,31-33,35-39 and 42-44* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to Applicant's submission filed 12/4/07, responding to the 9/10/07 Office action which detailed the rejection of claims 1, 3, 5-10, 12-18, 20-22, 31-33, 35-40, and 42-44. Claims 1, 6, 7, 16, 31, and 38 have been amended, claims 5, 18, and 40 have been canceled. Claims 1, 3, 6-10, 12-17, 20-22, 31-33, 35-39, and 42-44 remain pending in the application and have been fully considered by the examiner.

Response to Arguments/Amendments

2. On page 12 filed 12/4/07, Applicant does not argue in regard to the Double Patenting rejections, but has agreed to submit the necessary Terminal Disclaimers upon issuance of 10/784,492, or the instant application. As such, the rejection over 10/784,492 is maintained.

3. The amendment of claim 1 has overcome the rejection under 35 U.S.C. § 112, second paragraph. Likewise, this rejection is withdrawn.

4. Applicant's arguments filed 12/4/07 have been fully considered but they are not persuasive.

On page 15, Applicants argue that the cited art (WebLogic, Ringseth, and Monson-Haefel) "do not teach or suggest do not teach or suggest that the create(), etc. methods are indicated to a compiler as start, etc. methods by declarative annotations, as recited by amended claim 1." In particular, Applicants argue that Ringseth's declarative annotations "do not 'indicate to the compiler whether the identified method is at least one of a start method … wherein the start method applies to start of a stateful conversation between a client and the web service.'" However, Ringseth alone was not relied upon to teach these limitations. For example, WebLogic

is relied upon to disclose the generation of components based upon declarative annotations (See page 5 "Step 2" as cited in the following rejection). Ringseth teaches that declarative annotations within source code are useful for providing indications to a compiler for implementation of Web services (see paragraph [0032] on page 3). These teachings are not contended in Applicants remarks. Finally, Monson-Haefel is relied upon to teach declarative annotations which include start, continue, and finish methods used in assembling javabeans (see sections 7.4.2 and 10.6.3.2 as cited in the 9/10/07 rejection on page 14). The combination of references teaches the claimed limitations. Therefore, Applicants' argument is not persuasive.

In response to applicant's argument that there is no suggestion to combine the references (see bottom of page 15), the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988)and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). In this case, Monson-Haefel provides the suggestion that a stateful session bean acts on behalf of a client for its entire life cycle (see section 7.3 paragraph 1) so that changes to a "bean's state can affect the result of subsequent method invocations" (see section 7.3 paragraph 2).

At the bottom of page 15, Applicants argue that WebLogic, Ringseth, and Monson-Haefel teach away from the invention. However, Applicants have not shown where the references actually discourage using start, continue, or finish methods, or otherwise discourage the use of stateful components. To the contrary, Applicants appear to suggest that the references

are all interested in maintaining state (e.g. "compiling the methods and bean logic will result in the desired bean to maintain state"; also "[WebLogic, Ringseth, and Monson-Haefel] actually instantiate a persistent component (i.e., bean)"). In the absence of disclosure showing that a reference discourages state maintenance, the argument is not persuasive.

At the top of page 16, Applicants suggest that the limitations of claims 16, 31, and 38 are similar to those of claim 1 and patentable for the same reasons. However, it should be noted that claims 16 and 38 do not contain any recitation of a compiler, as does claim 1. Therefore, arguments in regard to a compiler are not persuasive in regard to claims 16 and 38.

At the top of page 20, in regard to the rejection of claim 15, Applicants argue that "Monson-Haefel does not, however, describe relations between a web service and an external service, much less assigning a unique identifier to a conversion with the external service, by a compiler, in response to declarative annotations." However, Monson-Haefel teaches instantiation of a conversation and return of an identifier upon invocation of a start method. See Section 7.4.2.1, e.g. "returns the EJB object's remote reference to the client." Weblogic further discloses generation of code by a compiler (see page 5 "Step 2") as addressed in the rejection of claim 1. Further limitations are addressed in the rejections of parent claims 1 and 13. Since the limitations appear to have been addressed, Applicants arguments are not persuasive.

Double Patenting

5. The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the "right to exclude" granted by a patent and to prevent possible harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686

F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970);and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to overcome an actual or provisional rejection based on a nonstatutory double patenting ground provided the conflicting application or patent is shown to be commonly owned with this application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37 CFR 3.73(b).

6. Claims 1, 3, 6-10, 12-17, 20-22, 31-33, 35-39, and 42-44 are provisionally rejected on the ground of nonstatutory obviousness-type double patenting as being unpatentable over claims 1-8, 19-23, 26, 27, 31-36, 38 and 39 of copending Application No. 10/784,492 (hereinafter "the '492 application") in view of "Enterprise JavaBeans" by Monson-Haefel (hereinafter "Monson-Haefel").

This is a provisional obviousness-type double patenting rejection.

1. A method of specifying a stateful web service within a procedural programming environment (see claim 19, page 5)*, the method comprising:*

first facilitating, by an integrated development environment of a computing device, a user in providing a source code representation of at least a portion of web service logic, the logic including one or more methods; See claim 1 on page 2 lines 3-4:

> an annotated source code, which is a programming language augmented with declarative meta-data capable of exposing program logic as a network-accessible service

Also see claim 5, e.g. "integrated development environment."

second facilitating, by the integrated development environment of the computing device, the user in identifying one of said one or more methods to be exposed as part of the stateful web service; See claim 1 on page 2 lines 5-6:

> at least one deployed service component capable of providing the network-accessible service to a client

Further, see claim 8 on page 3 lines 2-3:

> the annotated source code is capable of facilitating access to an external service, which can be one of stateful, stateless, synchronous, and asynchronous.

in response to user input, automatically specifying, by the integrated development environment of the computing device, one or more declarative annotations within the source code representation, the declarative annotations, when recognized by a compiler through analysis of the web service logic which includes the declarative annotations, causing the compiler to generate one or more persistent components to maintain conversational state related to the identified method. See claim 1 on page 2 lines 3-4:

> an annotated source code, which is a programming language **augmented with declarative meta-data**. [emphasis added]

Further, see claim 1 page 2 lines 5-10:

> an enhanced compiler capable of analyzing the **annotated source code**, recognizing numerous types of **meta-data annotations**, and generating a mechanism, which can include one or more of: object files, software components and deployment descriptors, to facilitate the deployment of the at least one service **component** [emphasis added]

Further, see claim 2 on page 2 lines 2-3:

> the system is capable of simultaneously managing multiple transactions, wherein each transaction can be a **conversation** of a request and/or a response from the client for the network-accessible service. [emphasis added]

The '492 application does not specifically claim limitations related to *start*, *continue*, or *finish* methods. However, Monson-Haefel teaches that "deployment descriptors" could be used as declarative annotations that indicate: *wherein the start*

method applies to the start of a stateful conversation between a client and the web service (see section 7.4.2.1), *the continue method applies to the continuation of an ongoing stateful conversation between a client and the web service* (see section 7.4.2 "ejbActivate()" on page 4 of section 7.4), *and the finish method applies to the completion of an ongoing stateful conversation between a client and the web service* (see section 7.4.2.3). Monson-Haefel further teaches that such methods can be entered as annotations in section 10.6.3.2. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Monson-Haefel's teaching of stateful sessions with the '492 application's compiler. One of ordinary skill would have been motivated to provide a dedicated stateful session bean to act on behalf of a client for its entire life cycle (see section 7.3 paragraph 1).

This is a provisional obviousness-type double patenting rejection because the conflicting claims have not in fact been patented.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claims 1, 6-8, 10, 16, 22, 31, 38 and 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over prior art of record "Using WebLogic Enterprise JavaBeans" by BEA Systems

(hereinafter "WebLogic") in view of U.S. PG-Pub. 2003/0014733 by Ringseth et al. (hereinafter "Ringseth") in view of "Enterprise JavaBeans" by Monson-Haefel (hereinafter "Monson-Haefel").

In regard to claim 1, WebLogic discloses:

A method of specifying a stateful web service within a procedural programming environment, (See page 5 steps 1-3) *the method comprising:*

first facilitating, by an integrated development environment of a computing device, a user in providing a source code representation of at least a portion of web service logic, the logic including one or more methods; See section III on page 4:

> There are three parts to using WebLogic EJB:
> 1. Develop an EJBean or obtain one from a third-party supplier.

Also see page 3 paragraph 5 for disclosure of methods in an EJB:

> ...an EJBean contains the business logic (**methods**)...

Further, WebLogic discloses "a framework for the development and deployment of EJBeans" by a user/developer (see Applicants' comments at the bottom of page 12, filed 4/19/07), and is interpreted as providing an integrated development environment.

second facilitating, by the integrated development environment of the computing device, the user in identifying one of said one or more methods to be exposed as part of the stateful web service; See page 2:

> Session beans (either **stateful** or stateless) [emphasis added]

Also see page 3, 4th paragraph:

> With the EJB model, you can write or buy business components (such as invoices, bank accounts and shipping routes) and, during **deployment** into a certain project, specify how the component

> should be used -- which users have **access to which methods**, whether the framework should automatically start a transaction or whether it should inherit the caller's transaction, and so on.
>
> [emphasis added]

Also page 6 "Step 2" discloses identification of methods to be exposed:

> Check the deployment descriptor and modify any of its properties for your particular deployment (if required).

in response to user input, automatically specifying, by the integrated development environment of the computing device, one or more declarative annotations within the source code representation, the declarative annotations, when recognized by a compiler through analysis of the <source code representation> which includes the declarative annotations, causing the compiler to generate one or more persistent components to maintain conversational state related to the identified method. See page 5 "Step 2":

> The **Deployment Descriptor** ties together the different classes and interfaces, and **is used to build the code-generated class files**. It also allows you to specify some aspects of the EJBean's **deployment** at runtime.
>
> ...WebLogic EJB includes a utility application DDCreator that **takes a text file specification and creates the appropriate serialized deployment descriptor**.
> [emphasis added]

Also see page 6 along with "Step 3":

> Generate the wrapper classes using the WebLogic EJB **compiler** (ejbc)...
> This will **create the appropriate files** for the bean...
> [emphasis added]

Also, page 8 discloses the general capabilities of EJBeans with respect to persistence and transactional, or "conversational", state:

> An entity EJBean can save its state in any **transactional** or non-transactional **persistent** storage...
> [emphasis added]

WebLogic does not expressly disclose "*one or more declarative annotations within the source code representation.*" However, Ringseth teaches the use of declarative annotations in source code to specify web services. See paragraph [0032]:

> In connection with the operation of an attribute provider in accordance with the invention, a compiler operates to implement SOAP-based Web services written according to the **declarative syntax** of the present invention. [emphasis added]

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Ringseth's declarative annotations with WebLogic's source code so that "a Web service developer may implement SOAP-based Web services without being required to understand the underlying details regarding the SOAP protocol, dispatching to the appropriate object and function, marshaling the XML, un-marshaling the XML, and generating the SOAP response" (see Ringseth paragraph [0032]).

WebLogic further discloses use of the "ejbCreate" function. See page 24, 4th paragraph. Also, using declarative annotations with a compiler is addressed above in reference to Weblogic. WebLogic and Ringseth do not expressly disclose specifics of the start, continue or finish methods. However, Monson-Haefel teaches that "deployment descriptors" could be used as declarative annotations that indicate: *wherein the start method applies to the start of a stateful conversation between a client and the web service* (see section 7.4.2.1), *the continue method applies to the continuation of an ongoing stateful conversation between a client and the web service* (see section 7.4.2 "ejbActivate()" on page 4 of section 7.4), *and the finish method applies to the completion of an ongoing stateful conversation between a client and the web service* (see section 7.4.2.3). Monson-Haefel further teaches that such methods can be entered as annotations in section 10.6.3.2. It would have been obvious to one of ordinary skill in the art at the

time the invention was made to use Monson-Haefel's teaching of stateful sessions with BAE Websphere's components. One of ordinary skill would have been motivated to provide a dedicated stateful session bean to act on behalf of a client for its entire life cycle (see section 7.3 paragraph 1) so that changes to a "bean's state can affect the result of subsequent method invocations" (see section 7.3 paragraph 2).

In regard to claim 6, the above rejection of claim 1 is incorporated. WebLogic and Ringseth do not expressly disclose: *wherein when a method declared to be a start method is invoked at run-time, a new instance of a conversation is created, and a unique identifier is associated with that conversational instance to facilitate management of multiple simultaneous conversations*. However, Monson-Haefel teaches instantiation of a conversation and return of an identifier upon invocation of a start method. See Section 7.4.2.1.

In regard to claim 7, the above rejection of claim 1 is incorporated. WebLogic and Ringseth do not expressly disclose: *wherein when a method declared to be a continue method or a finish method is invoked at run-time, a unique identifier provided by the client is obtained and used to access a corresponding instance of a conversation.* However, Monson-Haefel teaches the return of an identifier as noted in the above rejection of claim 6. Use of the representative identifier is inherent in referencing the session as discussed in section 7.4.2.3.

In regard to claim 8, the above rejection of claim 7 is incorporated. WebLogic and Ringseth do not expressly disclose: *wherein when a finish method is invoked at run-*

time, the corresponding instance of the conversation is destroyed after processing by the web service logic. However, Monson-Haefel teaches that the instance is destroyed after processing. See section 7.4.2.3.

In regard to claim 10, the above rejection of claim 1 is incorporated. WebLogic further discloses: *wherein the one or more declarative annotations are manually specified by a developer.* See page 6 "Step 2".

In regard to claim 16, WebLogic discloses:

In a procedural programming environment, a method of generating a stateful web service (See pages 6 and 7), *the method comprising:*

reading on one or more computing devices a segment of procedural source code representing at least a portion of the web service; parsing on one or more computing devices the <segment of source code> to identify the presence of one or more declarative annotations identifying an associated method within the segment as being stateful; generating on one or more computing devices one or more object codes defining one or more publicly accessible service components based at least in part upon the source code; See page 6 "Step 3":

> **Generate the wrapper classes** using the WebLogic EJB compiler (ejbc) with this command (typed on one line), **referencing the serialized deployment descriptor:**
> $ java weblogic.ejbc -d /weblogic/myserver/temp AccountBeanDD.ser
> ...
> This will create the appropriate files for the bean, and place them in a temporary directory

Reading and parsing source code is an inherent feature of a compiler, as object code could not be generated without both steps. This passage also shows use of a computing device by the invocation of a command that is "typed on one line".

generating on one or more computing devices meta-data based at least in part upon the one or more declarative annotations; associating on one or more computing devices meta-data with the one or more object codes; and See page 5 "Step 2":

> The Deployment Descriptor **ties together** the different classes and interfaces, and is **used to build** the code-generated class files.

if the presence of the one or more declarative annotations are identified by the parsing, generating, in response, on one or more computing devices one or more persistent components to maintain conversational state relating the associated method.

See page 5 "Step 2":

> The **Deployment Descriptor** ties together the different classes and interfaces, and **is used to build the code-generated class files.** It also allows you to specify some aspects of the EJBean's **deployment** at runtime.
>
> …WebLogic EJB includes a utility application DDCreator that **takes a text file specification and creates the appropriate serialized deployment descriptor.**
> [emphasis added]

Also see page 6 along with "Step 3":

> Generate the wrapper classes using the WebLogic EJB **compiler** (ejbc)…
> This will **create the appropriate files** for the bean…
> [emphasis added]

Page 8 discloses the general capabilities of EJBeans with respect to persistence and transactional, or "conversational", state:

> An entity EJBean can save its state in any **transactional** or non-transactional **persistent** storage…
> [emphasis added]

WebLogic does not expressly disclose a segment of source code containing declarative annotations. However, Ringseth teaches the use of declarative annotations in source code to specify web services. See paragraph [0032]:

> In connection with the operation of an attribute provider in accordance with the invention, a compiler operates to implement SOAP-based Web services written according to the **declarative syntax** of the present invention. [emphasis added]

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Ringseth's declarative annotations with WebLogic's source code so that "a Web service developer may implement SOAP-based Web services without being required to understand the underlying details regarding the SOAP protocol, dispatching to the appropriate object and function, marshaling the XML, un-marshaling the XML, and generating the SOAP response" (see Ringseth paragraph [0032]).

All further limitations of claim 16 have been addressed in the above rejection of claim 1, and would be obvious for the same reasons.

In regard to claim 22, the above rejection of claim 16 is incorporated. WebLogic further discloses: *wherein the source code is written in the Java programming language.* WebLogic discloses implementation using Enterprise JavaBeans (EJB - See page 2) which is an API that uses the Java programming language.

In regard to claim 31, WebLogic discloses:

a storage medium; and a plurality of programming instructions stored on the storage medium (page 7 step 5 shows a directory path for storage of programming instructions), *to provide an integrated development environment to facilitate a user in providing input associated with web service logic of a stateful web service...* WebLogic discloses "a framework for the development and deployment of EJBeans" by a user/developer (see Applicants' comments at the bottom of page 12, filed 4/19/07), and is interpreted as providing an integrated development environment.

and automatically specify, in response to the user input, one or more declarative annotations ..., the declarative annotations associated with an identified method of the web service logic page 2:

> Session beans (either **stateful** or stateless) [emphasis added]

Also page 6 "Step 2" discloses identification of methods to be exposed:

> Check the deployment descriptor and modify any of its properties for your particular deployment (if required).

the declarative annotations, when recognized by a compiler through analysis of the web service logic, causing the compiler to generate one or more persistent components to maintain conversational state related to the identified method. See page 5 "Step 2":

> The **Deployment Descriptor** ties together the different classes and interfaces, and **is used to build the code-generated class files**. It also allows you to specify some aspects of the EJBean's **deployment** at runtime.
>
> ...WebLogic EJB includes a utility application DDCreator that **takes a text file specification and creates the appropriate serialized deployment descriptor**.
> [emphasis added]

Also see page 6 along with "Step 3":

> Generate the wrapper classes using the WebLogic EJB **compiler** (ejbc)...
> This will **create the appropriate files** for the bean...
> [emphasis added]

WebLogic does not expressly disclose "*declarative annotations within the web service logic.*" However, Ringseth teaches the use of declarative annotations in source code to specify web services. See paragraph [0032]:

> In connection with the operation of an attribute provider in accordance with the invention, a compiler operates to implement SOAP-based Web services written according to the **declarative syntax** of the present invention. [emphasis added]

It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Ringseth's declarative annotations with WebLogic's source

code so that "a Web service developer may implement SOAP-based Web services without being required to understand the underlying details regarding the SOAP protocol, dispatching to the appropriate object and function, marshaling the XML, un-marshaling the XML, and generating the SOAP response" (see Ringseth paragraph [0032]).

All further limitations of claim 31 have been addressed in the above rejection of claim 1, and would be obvious for the same reasons.

In regard to claim 38, WebLogic discloses: *An article of manufacture comprising: a storage medium having stored therein a plurality of programming instructions.* Page 7 step 5 shows a directory path for storage of programming instructions as cited in the rejection of claim 31. All further limitations were addressed in the above rejection of claims 1 and 16, and would be obvious for the same reasons.

In regard to claim 44, the above rejection of claim 38 is incorporated. The cited art of claim 38 does not expressly disclose the features of claim 44. However, all further limitations have been addressed in the above rejection of claim 22, and would be obvious for the same reasons.

9. Claim 3 is rejected under 35 U.S.C. 103(a) as being unpatentable over prior art of record WebLogic, Ringseth, and Monson-Haefel as applied to claim 1 above, and further in view of prior art of record "EJBDoclet", December 21 2000, by dreamBean Software (hereinafter "dreamBean").

In regard to claim 3, the above rejection of claim 1 is incorporated. The cited art of claim 1 does not expressly disclose declarative annotations within a comment field preceding an identified method. However, dreamBean teaches a tool for generating "EJB files from a commented bean source-file" (page 1 paragraph 1). dreamBean further teaches using comment fields preceding a method to support generation of externally accessible methods. See middle of page 5, where the annotation "@remote-method" appears in a comment preceding the identified methods "deposit" and "withdraw". It would have been obvious to one of ordinary skill in the art at the time the invention was made to use dreamBean's declarative annotation with WebLogic's source code. One of ordinary skill would have been motivated to automatically generate a remote interface (see dreamBean page 1 under "Features").

10. Claim 9, 17, and 39 are rejected under 35 U.S.C. 103(a) as being unpatentable over WebLogic, Ringseth, and Monson-Haefel as applied to claim 1 above, and further in view of prior art of record U.S. Patent 5,812,768 to Pagé et al. (hereinafter "Page").

In regard to claim 9, the above rejection of claim 1 is incorporated. The cited art of claim 1 does not expressly disclose: *wherein the one or more declarative annotations indicate to the compiler whether the identified method is buffered, wherein if the identified method is buffered the compiler instantiates one or more queues to temporarily store one or more requests for the identified method.* However, Page teaches that

interaction with web services can be implemented as buffered messages that operate asynchronously via message queues. See column 6 lines 39-46. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use Page's queues with WebLogic's methods. One of ordinary skill would have been motivated to implement "store and forward" technology in order to provide reliable data delivery as suggested by Page (see column 2 lines 30-34).

In regard to claim 17, the above rejection of claim 16 is incorporated. The cited art of claim 16 does not expressly disclose the features of claim 17. However, all further limitations have been addressed in the above rejection of claim 9, and would be obvious for the same reasons.

In regard to claim 39, the above rejection of claim 38 is incorporated. The cited art of claim 38 does not expressly disclose the features of claim 39. However, all further limitations have been addressed in the above rejection of claim 9, and would be obvious for the same reasons.

11. Claims 12 and 32 are rejected under 35 U.S.C. 103(a) as being unpatentable over WebLogic, Ringseth, and Monson-Haefel as applied to claims 1 and 31 above, and further in view of U.S. Patent 6,230,160 to Chan et al. (hereinafter "Chan").

In regard to claim 12, the above rejection of claim 1 is incorporated. The cited art of claim 1 does not expressly disclose: *wherein said input includes graphical manipulation of the identified method by the developer via the integrated development*

environment. However, in an analogous environment, Chan teaches an IDE for manipulation of program elements and methods. See FIG. 4A and column 8 lines 19-30.

In regard to claim 32, the above rejection of claim 31 is incorporated. The cited art of claim 31 does not expressly disclose the features of claim 32. However, all further limitations have been addressed in the above rejections of claims 1 and 12, and would be obvious for the same reasons.

12. Claims 13, 15, 20, 21, 35, 37, 42 and 43 are rejected under 35 U.S.C. 103(a) as being unpatentable over WebLogic, Ringseth, and Monson-Haefel as applied to claim 1 above, and further in view of the "Background of the Invention" section appearing on pages 1-3 of the originally filed specification (hereinafter "BOTI").

In regard to claim 13, the above rejection of claim 1 is incorporated. The cited art of claim 1 does not expressly discloses: *a proxy object designed to facilitate interaction by the web service with one of an external web service or client*. However, BOTI teaches implementation of proxy objects. See page 2 lines 17-19. It would have been obvious to one of ordinary skill in the art at the time the invention was made to use BOTI's teaching of implementation of a proxy object with WebLogic's web service. One of ordinary skill would have been motivated to automatically generate a required proxy mechanism.

In regard to claim 15, the above rejection of claim 13 is incorporated. The cited art of claim 13 does not expressly disclose the features of claim 15. However, all further

limitations have been addressed in the above rejection of claim 6, and would be obvious for the same reasons.

In regard to claim 20, the above rejection of claim 16 is incorporated. The cited art of claim 16 does not expressly disclose the features of claim 20. However, all further limitations have been addressed in the above rejection of claim 13, and would be obvious for the same reasons.

In regard to claim 21, the above rejection of claim 20 is incorporated. The cited art of claim 20 does not expressly disclose the features of claim 21. However, all further limitations have been addressed in the above rejection of claim 6, and would be obvious for the same reasons.

In regard to claim 35, the above rejection of claim 31 is incorporated. The cited art of claim 31 does not expressly disclose the features of claim 35. However, all further limitations have been addressed in the above rejection of claim 13, and would be obvious for the same reasons.

In regard to claim 37, the above rejection of claim 35 is incorporated. The cited art of claim 35 does not expressly disclose the features of claim 37. However, all further limitations have been addressed in the above rejection of claim 15, and would be obvious for the same reasons.

In regard to claim 42, the above rejection of claim 38 is incorporated. The cited art of claim 38 does not expressly disclose the features of claim 42. However, all further limitations have been addressed in the above rejection of claim 20, and would be obvious for the same reasons.

In regard to claim 43, the above rejection of claim 42 is incorporated. The cited art of claim 42 does not expressly disclose the features of claim 43. However, all further limitations have been addressed in the above rejection of claim 21, and would be obvious for the same reasons.

13. Claims 14 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over WebLogic, Ringseth, Monson-Haefel and BOTI as applied to claims 13 and 35 above, and further in view of Page.

In regard to claim 14, the above rejection of claim 13 is incorporated. The cited art of claim 13 does not expressly disclose the features of claim 14. However, all further limitations have been addressed in the above rejection of claim 9, and would be obvious for the same reasons.

In regard to claim 36, the above rejection of claim 35 is incorporated. The cited art of claim 13 does not expressly disclose the features of claim 14. However, all further limitations have been addressed in the above rejection of claim 14, and would be obvious for the same reasons.

14. Claim 33 is rejected under 35 U.S.C. 103(a) as being unpatentable over WebLogic, Ringseth, Monson-Haefel, and Chan as applied to claim 32 above, and further in view of dreamBean.

In regard to claim 33, the above rejection of claim 32 is incorporated. The cited art of claim 32 does not expressly disclose the features of claim 33. However, all further

limitations have been addressed in the above rejection of claim 3. It would have been obvious to one of ordinary skill in the art at the time the invention was made to combine WebLogic with dreamBean for the same reasons presented in the rejection of claim 3.

Conclusion

15. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

U.S. PG-Pub 2005/0021594 A1 by Bernardin et al. discloses a computing environment that provides proxy objects associated with asynchronous interfaces for communication with clients. See paragraph [0019]. This appears to be similar to the subject matter of claims 35-37.

16. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

17. Any inquiry concerning this communication or earlier communications from the examiner should be directed to J. Derek Rutten whose telephone number is (571)272-3703. The examiner can normally be reached on M-F 9:00-5:30.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571)272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/jdr/

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192